Regression Planning

Idea: search backwards from the goal description: nodes correspond to subgoals, and arcs to actions.
A subgoal is an assignment of values to some features.

Search problem:

- Nodes are subgoals
- There is an arc $\langle g, g' \rangle$ labeled with action A if
  - A achieves one of the assignments in g
  - g' is a proposition that must be true immediately before action A so that g is true immediately after.
- The start node is the goal to be achieved.
- $goal(g)$ is true if g is a proposition that is true of the initial state.

Defining nodes and arcs

- A node g can be represented as a set of assignments of values to variables:

  $$[X_1 = v_1, \ldots, X_n = v_n]$$

  This is a set of assignments you want to hold.
- The last action achieves one of the $X_i = v_i$, and does not achieve $X_j = v'_j$ where v'_j is different to v_j.
- The neighbor of g along arc A must contain:
  - The prerequisites of action A
  - All of the elements of g that were not achieved by A

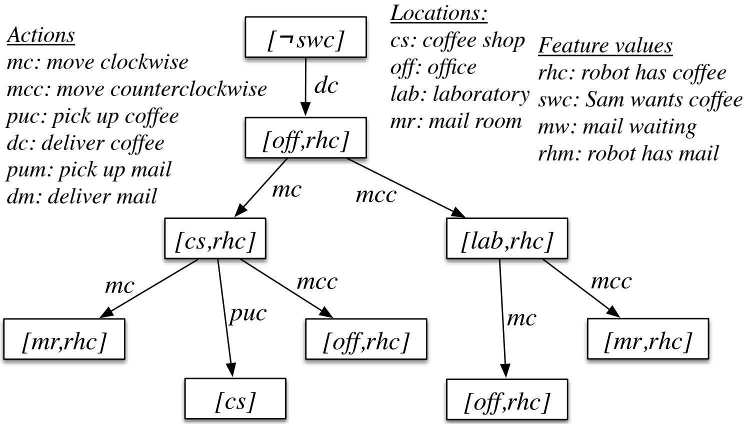
  it must be consistent = have at most one value for each feature.

Formalizing arcs using STRIPS notation

$\langle g, g' \rangle$ is an arc labeled with action A where g is $[X_1 = v_1, \ldots, X_n = v_n]$ and A is an action, if

- $\exists i \; X_i = v_i$ is on the effects list of action A
- $\forall j \; X_j = v_j'$ is not on the effects list for A, where $v_j' \neq v_j$
- $g' = \mathit{preconditions}(A) \cup \{X_k = v_k \in g : X_k = v_k \notin \mathit{effects}(A)\}$ if it is consistent

Regression example



Actions
- *mc: move clockwise*
- *mcc: move counterclockwise*
- *puc: pick up coffee*
- *dc: deliver coffee*
- *pum: pick up mail*
- *dm: deliver mail*

Locations:
- *cs: coffee shop*
- *off: office*
- *lab: laboratory*
- *mr: mail room*

Feature values
- *rhc: robot has coffee*
- *swc: Sam wants coffee*
- *mw: mail waiting*
- *rhm: robot has mail*

Loop detection and multiple-path pruning

- Goal G_1 is simpler than goal G_2 if G_1 is a subset of G_2.
  - It is easier to solve $[cs]$ than $[cs, rhc]$.
- If you have a path to node N have already found a path to a *simpler* goal, you can prune the path N.

Improving Efficiency

- You can define a heuristic function that estimates how difficult it is to solve a goal from a state.
  A heuristic function defined the cost of getting from a state to a (sub)goal. This is the same as a heuristic for the forward planner.
- You can use domain-specific knowledge to remove impossible goals, e.g.
  - It is often not obvious from an action description to conclude whether an agent can hold multiple items at any time.

Comparing forward and regression planners

- Which is more efficient depends on:
  - The branching factor
  - How good the heuristics are
- Forward planning is unconstrained by the goal (except as a source of heuristics).
- Regression planning is unconstrained by the initial state (except as a source of heuristics)